Strategies for Parallel Implicit Monte Carlo (U)

by

Todd Urbatsch, XTM

Los Alamos National Laboratory
Los Alamos National, NM 87545

Outline

1. Problem Statement
2. General Considerations
3. Three Basic Parallelization Schemes
4. 1-D Hypothetical Examples
5. 2-Step Parallelization Schemes
6. Radiation-Hydrodynamics Considerations
7. Summary

Abstract

Parallel Monte Carlo methods are successful because particles are typically independent and easily distributed to multiple processors. For distributed memory, each processor must have enough memory to hold the entire mesh. Unfortunately, three-dimensional problems with fine resolution may easily exceed the available memory and necessitate some sort of spatial decomposition of the problem. We present three basic schemes, which represent parallelization strictly in particles, strictly in space, and in both space and particles. We propose a two-step scheme that is based on a scheme proposed by Lawrence Livermore National Laboratory (LLNL). Our scheme has potential for larger speedups over the basic schemes and LLNL's two-step scheme.

Summary

We present our strategy for parallelizing Implicit Monte Carlo (IMC) calculations. The IMC algorithm, developed by Fleck and Cummings, is linearized, meaning that our parallelization strategy applies to any other linear Monte Carlo algorithm as well. Traditionally, Monte Carlo codes are made parallel by repeating the whole mesh on every processor and splitting the particles between the processors. This scheme is not possible when the entire mesh will not fit on each processor or even a group of processors. In this case, some sort of domain decomposition is necessary, and parallelization becomes non-trivial.

We first present three basic schemes for parallelization: full replication, full domain decomposition, and general domain decomposition/replication. We look at two hypothetical, one-dimensional examples that compare the full domain decomposition scheme and the general scheme. Next, we present a scheme proposed by Lawrence Livermore National Laboratory (LLNL), which can be generalized to a two-step scheme and can be viewed as virtual full replication on subsets of processors. We then present our two-step scheme, which is based upon LLNL's two-step scheme, but appears to have potential for higher speedups. We conclude with some radiation-hydrodynamics considerations.

Problem Statement

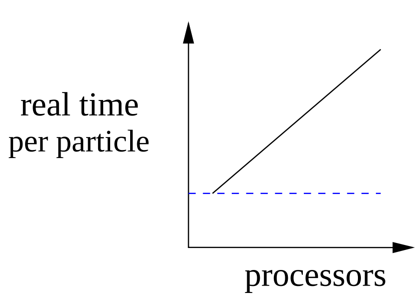
How do we parallelize

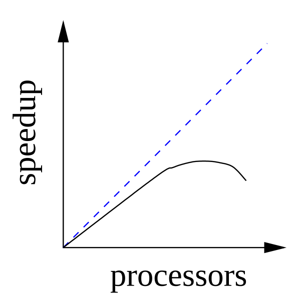
- **large**,
- **highly-resolved**

Monte Carlo calculations, especially with **limited-memory constraints**?

The parallelization should **minimally**

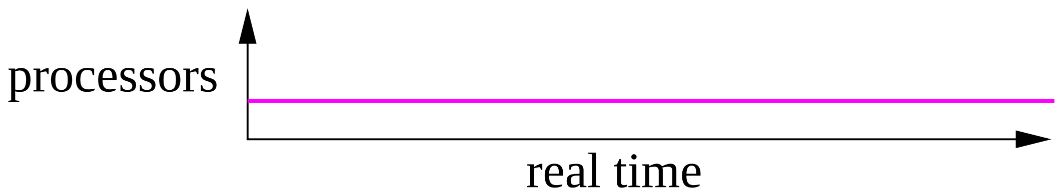
- decrease raw speed, and
- increase unparallelizable overhead.



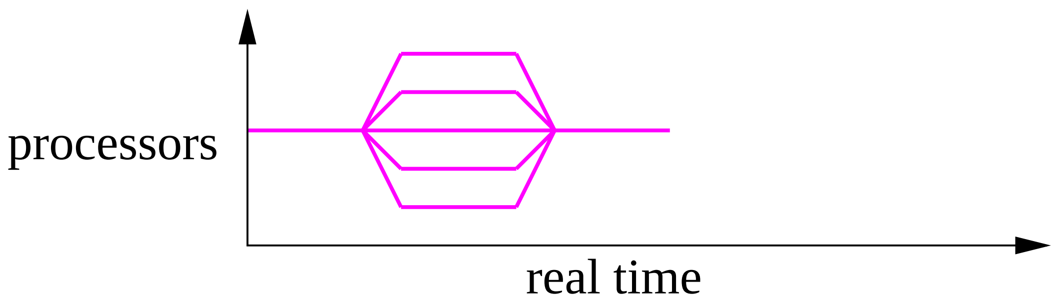


Serial and Parallel Runtimes

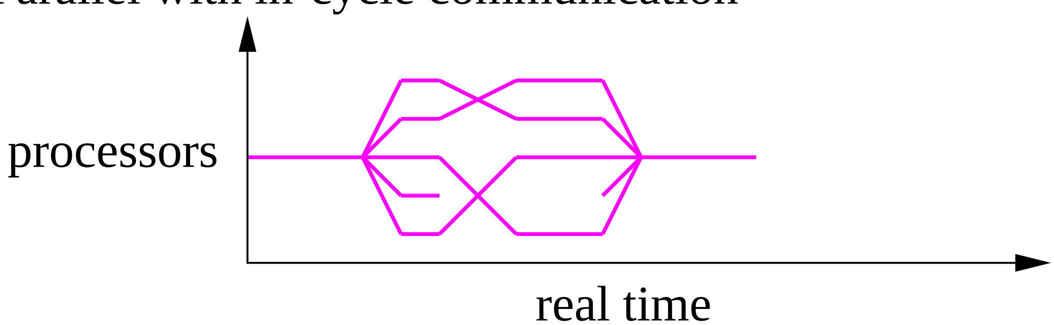
Serial



Parallel with NO in-cycle communication



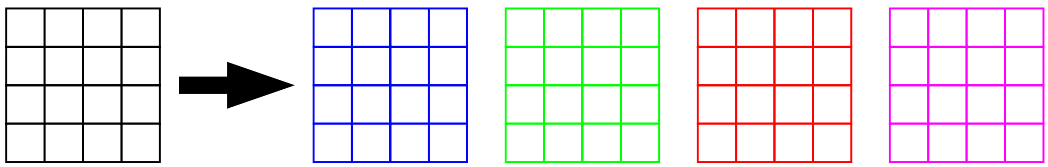
Parallel with in-cycle communication



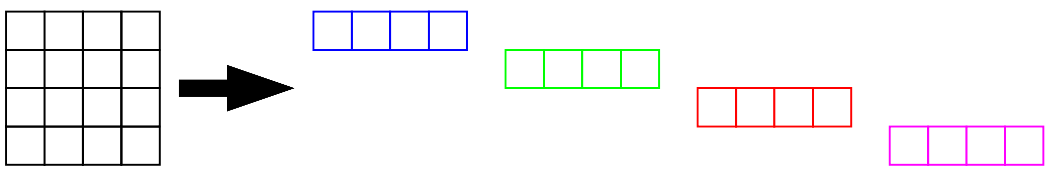
$$T = T_{un} + T_{overhead} + T_{work} + T_{in\text{-}cyc}$$

Basic Parallelization Schemes

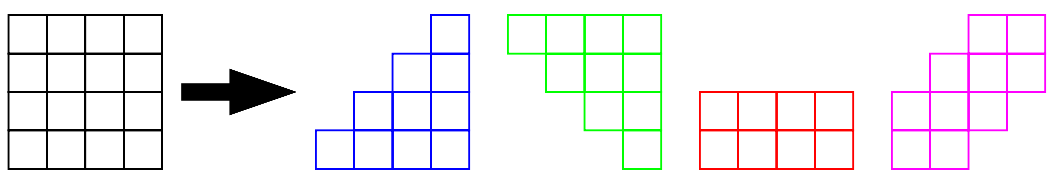
Full Replication



Full Domain Decomposition



General Domain Decomposition/Replication



Limits to	Full DD	as processor capacity	decreases
	Full Repl.		increases

Load Balancing

time–explicit

Replicate and distribute cells according to

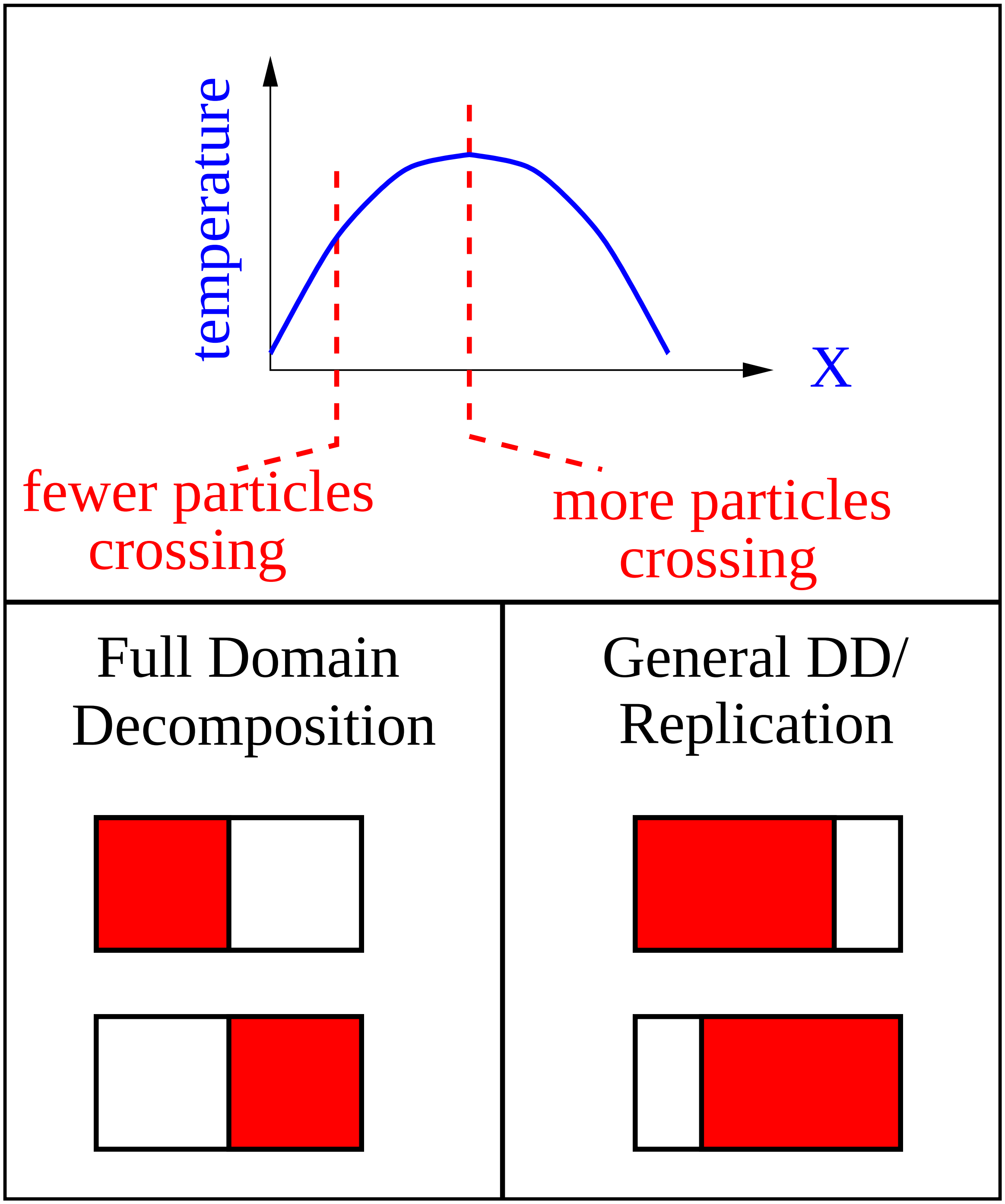
- number of source particles in cell

time–implicit

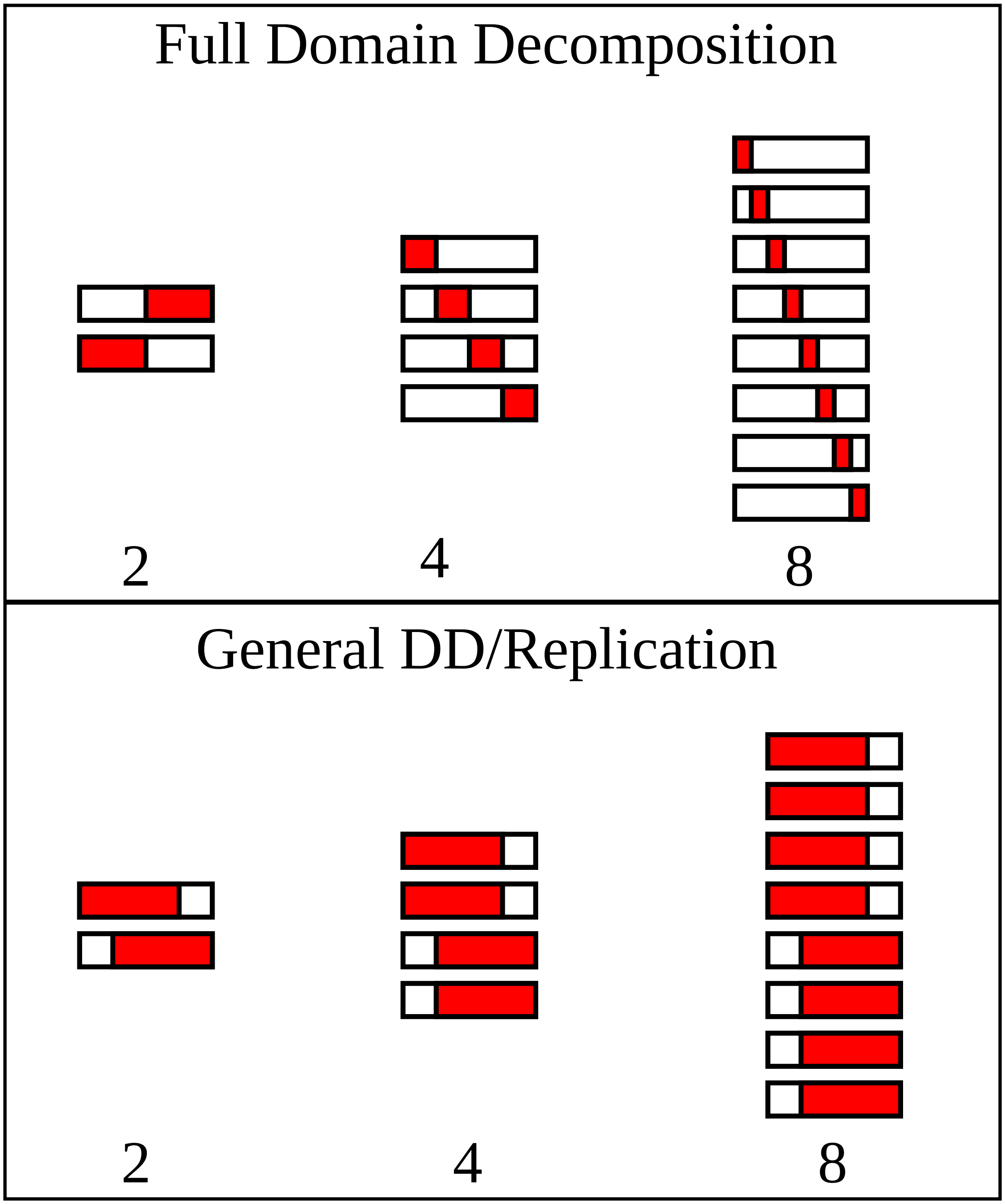
Replicate and distribute a cell according to

- number of source particles in cell
- work per particle in cell
- cell's optical– and time–proximity to sources

Domain Boundary Blurring

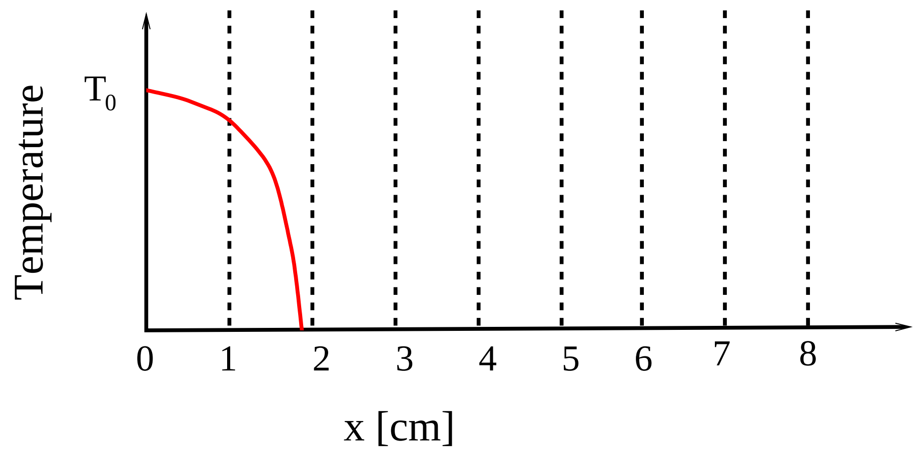


Scaling with Processors

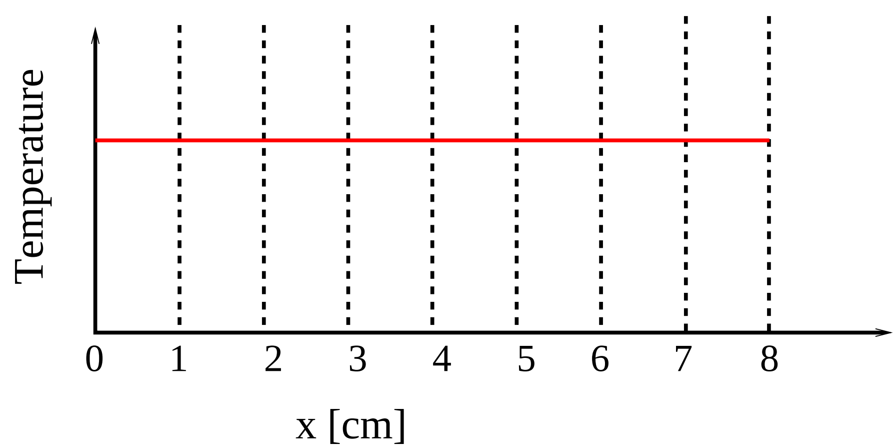


1-D Examples

Marshak Wave

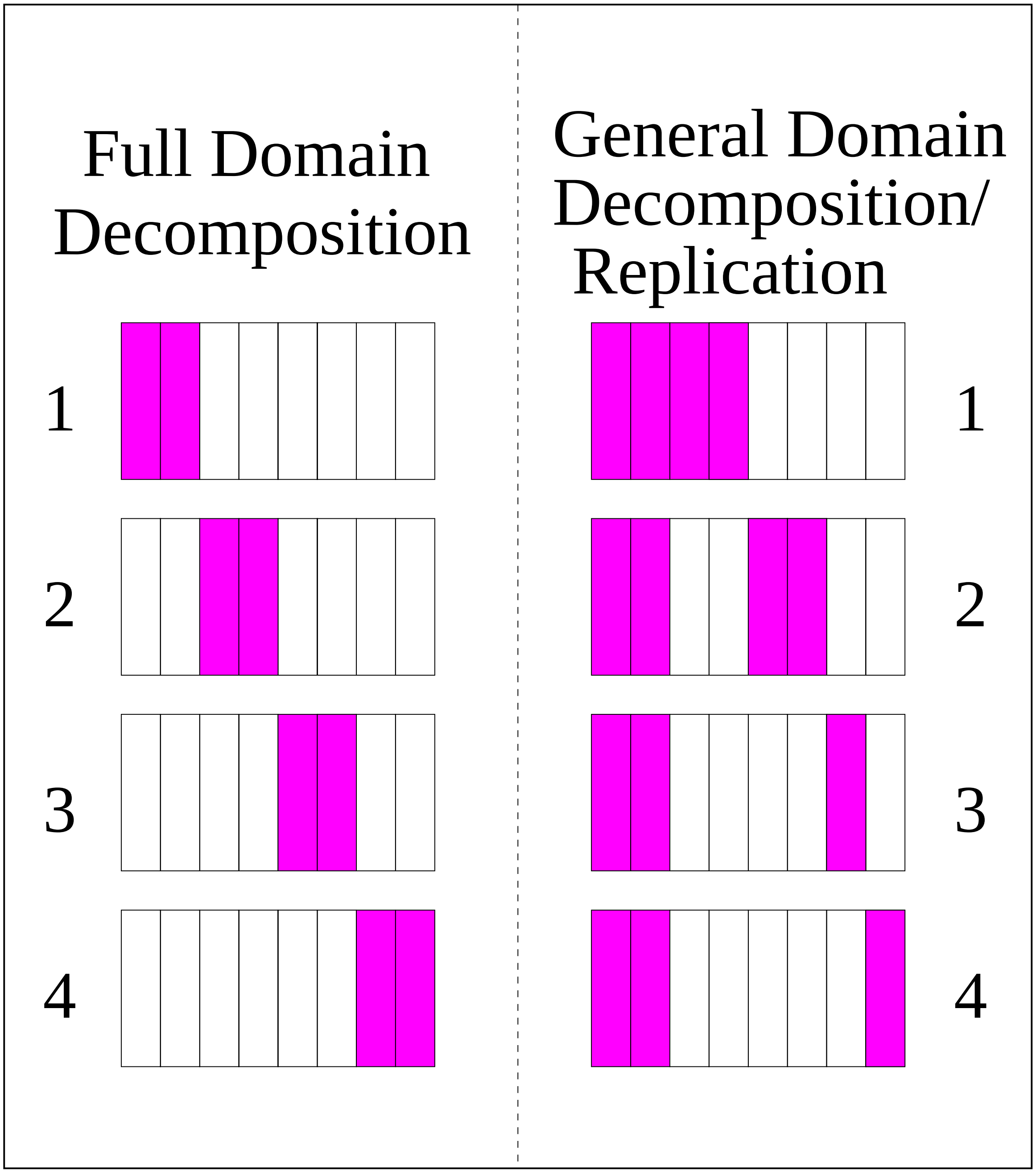


Flat Distribution

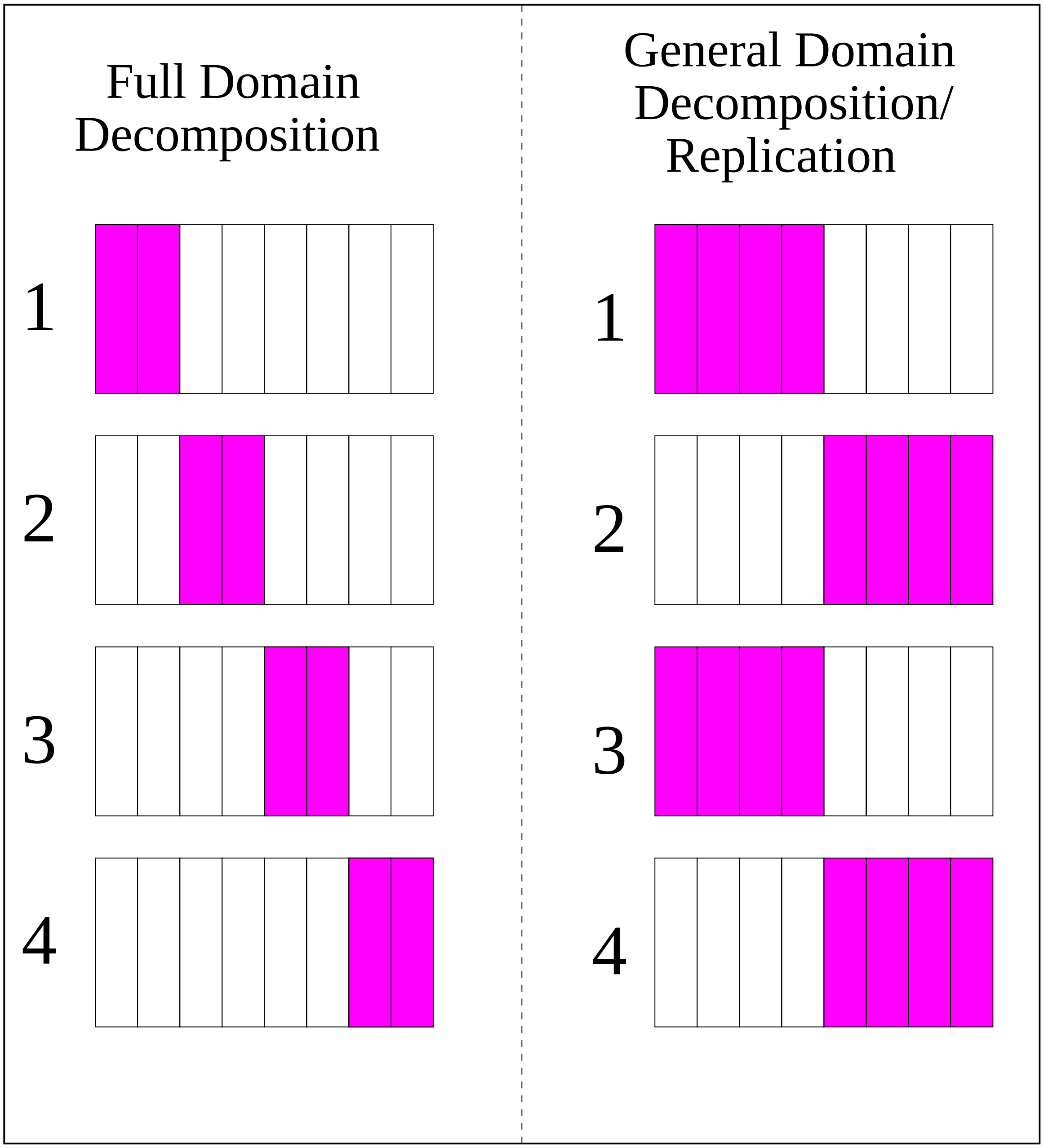


Assume Processor Capacity = 4 cells

Marshak Topologies



Flat Topologies



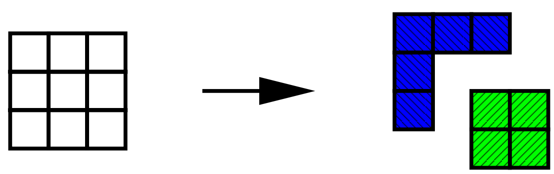
LLNL's Two-Step Scheme

Have P processors: ○ ○ ○ ○ ○ ○ ○ ○

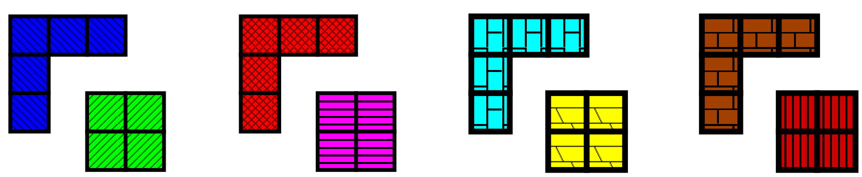
Divide into $S=P/P_{set}$ sets of P_{set} processors:

(○ ○) (○ ○) (○ ○) (○ ○)

1) Perform a Full DD on P_{set} processors.



2) Replicate the subset S total times.



3) Limit communication to within set.

- (work+comm) replicated S times.
- Only S replication speedup possible.

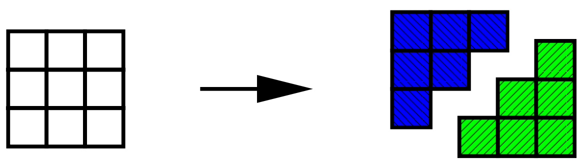
Our Two-Step Scheme

Have P processors: ○ ○ ○ ○ ○ ○ ○ ○

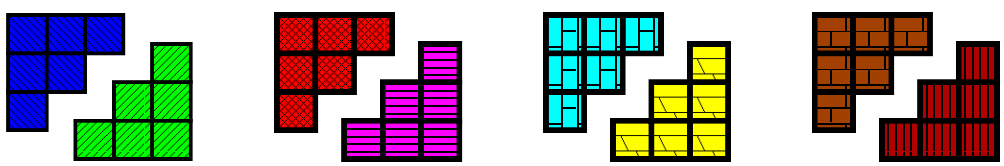
Divide into $S=P/P_{set}$ sets of P_{set} processors:

(○ ○) (○ ○) (○ ○) (○ ○)

1) Perform General DD/R on P_{set} procs.



2) Replicate the subset S total times.

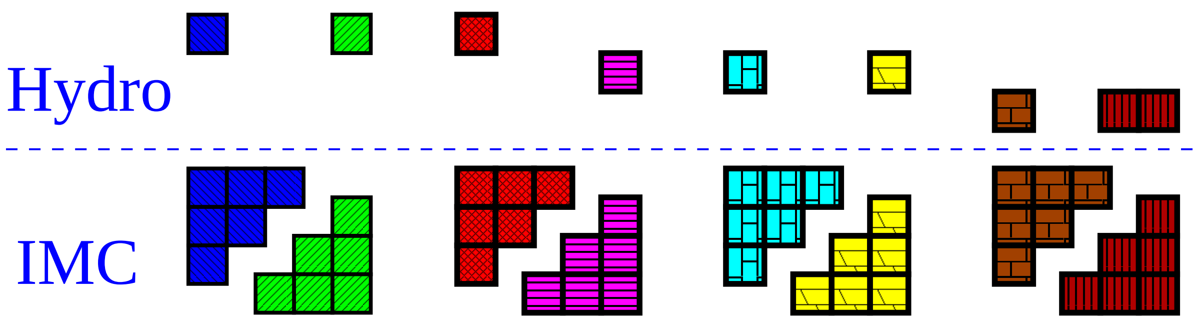


3) Limit communication to within set.

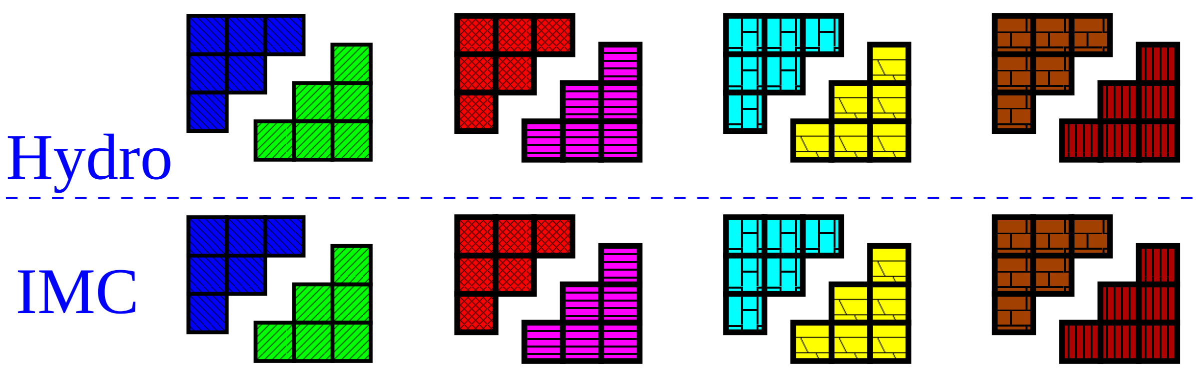
- Higher cost of General DD/Replication communication is limited to subset.
- (work+comm) replicated S times.
- Localized full P speedup possible.

Rad-Hydro Considerations

Full DD Hydrodynamics: Full Remap



Consistent-Mesh Hydrodynamics



Summary

- 3 Basic Parallelization Schemes
- 1-D Examples
- 2-Step Parallelization Schemes
- Radiation-Hydrodynamics Considerations